

02986.P008

UNITED STATES PATENT APPLICATION  
FOR  
METHODS AND APPARATUSES FOR DESIGNING INTEGRATED  
CIRCUITS

INVENTORS:

KENNETH S. McELVAIN  
ROBERT ERICKSON

PREPARED BY:

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN  
12400 WILSHIRE BOULEVARD  
SEVENTH FLOOR  
LOS ANGELES, CA 90025-1026

(408) 720-8598

**EXPRESS MAIL CERTIFICATE OF MAILING**

"Express Mail" mailing label number: EL 164 803 997 US

Date of Deposit: May 17, 1999

I hereby certify that I am causing this paper or fee to be deposited with the United States Postal Service  
"Express Mail Post Office to Addressee" service on the date indicated above and that this paper or fee has  
been addressed to the Assistant Commissioner for Patents, Washington, D. C. 20231

Claire Walters  
(Typed or printed name of person mailing paper or fee)

Claire Walters  
(Signature of person mailing paper or fee)

May 17, 1999  
(Date signed)

## METHODS AND APPARATUSES FOR DESIGNING INTEGRATED CIRCUITS

### FIELD OF THE INVENTION

5           The present invention relates generally to the field of designing integrated circuits, and more particularly to the design of integrated circuits through a synthesis process which begins with the use of a hardware description language.

### BACKGROUND OF THE INVENTION

10           For the design of digital circuits on the scale of VLSI (very large scale integration) technology, designers often employ computer aided techniques. Standard languages such as Hardware Description Languages (HDLs) have been developed to describe digital circuits to aide in the design and simulation of complex digital circuits. Several hardware description languages, such as VHDL and Verilog, have evolved as  
15 industry standards. VHDL and Verilog are general purpose hardware description languages that allow definition of a hardware model at the gate level, the register transfer level (RTL) or the behavioral level using abstract data types. As device technology continues to advance, various product design tools have been developed to adapt HDLs for use with newer devices and design styles.

20           After writing and compiling HDL code, the design of an integrated circuit (IC) or a system which includes multiple integrated circuits must be verified to be correct. Verification is often the biggest bottle-neck for today's complex circuit designs, such as ASICs (Application Specific Integrated Circuits). Continually advancing processing technology, and the corresponding explosion in design size and

complexity, have led to verification problems that are impossible or difficult to solve using traditional simulation tools. In addition, an increasing number of circuit applications that process large quantities of data (e.g. video display) in real time require verification techniques that run at near real time speeds. As a result, a growing

5 number of ASIC designers are building boards (typically printed circuit boards) using multiple ICs known as field programmable gate arrays (FPGA) to verify their ASIC designs. Another advantage of building prototype boards with FPGAs is that it allows software development and debugging to happen in parallel with the design and debug of the final ASIC thereby reducing the development time of the product.

10 However, ASIC designers face several challenges in prototyping and debugging the design using FPGAs. One of the biggest challenges is the capacity of even the largest FPGAs is much smaller than the size of a complex ASIC. This means that the designers must struggle to partition their design into multiple FPGAs, with few or no tools to help them make good partitioning decisions, and no way to model the

15 characteristics during synthesis of partitioning. Thus, designers must iterate between an HDL synthesis process and partitioning of the board implementation which requires a considerable amount of time and tends to lengthen the product development time.

Two approaches are known in the prior art for creating/partitioning a design

20 across multiple integrated circuits (ICs). One approach partitions a design after a completed synthesis from HDL code. This approach is shown in **Figure 1A**. Another approach separately synthesizes the HDL codes that had been designed/written for separate integrated circuits; this is shown in **Figure 1B**.

**Figure 1A** shows a method 10 for partitioning a design across multiple ICs after HDL synthesis is completed. In operation 12 of this method, HDL code is prepared with no attempt at partitioning logic between multiple ICs. In operation 14, the HDL prepared in operation 12 is compiled to produce a netlist which is typically optimized by performing logic optimization. Thereafter, a mapping process maps the netlist to a specific target technology/architecture which is determined by the targeted architecture of the FPGA. It is well-known that the various vendors of FPGA ICs such as Xilinx and Altera, use different arrangements of transistors to create logic circuits. Accordingly, a technology independent netlist that is often created by the compilation of HDL code must be mapped to the specific technology in the vendor's IC which will be used to implement the logic. At the end of operation 14, the synthesis has been completed and a netlist which is specific to the technology/architecture used in the vendor's IC is now provided. This netlist is effectively at a gate level and is partitioned in operation 16 between/among several ICs. After operation 16 which partitions to create several ICs, a conventional place and route operation is performed on the logic circuit in operation 18 in order to create a design of the circuitry in each of the ICs.

An example of the approach shown in **Figure 1A** is a logic emulation product System Realizer by Quickturn. Another example of this approach is provided by a software tool from Auspey of Cupertino, California. The quality of the resulting partitioning is poor, requiring either a large number of integrated circuits at low utilization on each IC or a substantial manual effort to guide the partitioning. This

manual effort is difficult is because of a tremendous level of detail in the gate level netlist.

**Figure 1B** shows another example in the prior art which is used to create separate ICs. This approach attempts to design the HDL source code, before HDL synthesis, into separate HDL designs which are connected by a netlist. In this method 5 20, operation 22 involves the preparation of HDL code for a first integrated circuit while operation 24 involves the preparation of HDL code for a second integrated circuit. Then a netlist in operation 26 is prepared (e.g., by preparing another HDL code) which will connect the two integrated circuits through a structure with 10 connections, such as connections on a printed circuit board (PCB). Then in operation 28, synthesis is performed separately for each of the two integrated circuits. Thus, the first integrated circuit is synthesized by operations 30, 32, and 34, and the second integrated circuit is synthesized by operations 36, 38, and 40. While the quality of the resulting partition can be good from method 20, this method often requires substantial 15 manual efforts in creating and changing the HDL code to reorganize design hierarchy such that two separate chips are synthesized from the two HDL files.

From the foregoing, it can be seen that HDL synthesis and partitioning are typically separate, disjoint steps in the process flow. Synthesis decisions are made without understanding how the design is to be partitioned across multiple ICs, such as 20 FPGAs in the case of method 10, which results in prototypes that do not run at the desired speed. Further, partitioning decisions are made without any feedback about their impact on utilization and system performance. The result is that the designers must typically iterate several times between synthesis and partitioning. Each iteration

is time consuming and extremely tedious, resulting in loss of productivity and increasing the time to develop a prototype. This process may also need to be repeated when the prototype has been exercised (tested) and changes are made to the original source code. Thus, it can be seen that it is desired to provide an improved method and

5 apparatus for designing integrated circuits.

## SUMMARY OF THE INVENTION

The present invention discloses methods and apparatuses for designing an integrated circuit. According to one aspect of the present invention, an exemplary method for designing a plurality of integrated circuits partitions a technology  
5 independent RTL (register transfer level) netlist between the plurality of integrated circuits. In a typical example of this method, a hardware description language (HDL) code is compiled to produce the technology independent RTL netlist after the compilation. The partitioning is then performed before any mapping of the technology independent RTL netlist to a selected technology/architecture.

10 According to another aspect of the present invention, an exemplary method compiles an HDL code to produce an RTL netlist representation which specifies a plurality of ICs. Logic designed for placement on one of the plurality of ICs is selected and replicated for placement on another one of the plurality of ICs. In a typical example of this method, a partitioning operation may be used to specify the  
15 plurality of integrated circuits and then a replication operation may be performed to replicate certain selected logic from one chip onto another chip.

Another aspect of the present invention involves the splitting of one RTL component into a plurality of RTL components. An exemplary method according to this aspect compiles an HDL code to produce an RTL netlist representation. One RTL  
20 component in this netlist representation is selected and split into a first RTL component which is designed for placement onto a first integrated circuit and a second RTL component which is designed for placement onto a second integrated circuit.

The present invention also discloses apparatuses, including software media which may be used to design integrated circuits. For example, the present invention includes digital processing systems which are capable of designing integrated circuits according to the present invention, and the invention also provides machine readable  
5 media which, when executed on a digital processing system, such as a computer system, causes the digital processing system to execute a method for designing integrated circuits.

Other features of the present invention will be apparent from the accompanying drawings and from the detailed description which follows.



## BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example and not limitation in the figures of the accompanying drawings in which like references indicate similar elements.

5           **Figure 1A** and **Figure 1B** show two methods in the prior art for designing integrated circuits.

**Figure 2** is a block diagram of a digital processing system that may be used to implement embodiments of the present invention.

10           **Figure 3** is a flow chart illustrating operations of an HDL synthesis process that is used with embodiments of the present invention.

**Figure 4A** is a flow chart illustrating one particular method of the present invention in which partitioning is performed within the synthesis process.

**Figure 4B** shows an exemplary method of the present invention in which floor planning is performed within a synthesis process.

15           **Figure 5A** shows an example of an optional process performed in the methods shown in the **Figures 4A** and **4B**.

**Figure 5B** shows an example of a hierarchical interconnect optimization technique.

20           **Figure 6** shows an example of an optional method for performing hierarchical resource estimation which may be performed in the methods shown in **Figures 4A** and **4B**.

**Figure 7A** shows an example of a partitioning operation according to one aspect of the present invention.

**Figure 7B** shows an example of a user interface which allows for a partitioning to be used according to one aspect of the present invention.

**Figure 8A** is an example of HDL source code which itself is hierarchical.

**Figure 8B** is an example of a top level RTL netlist resulting from the  
5 compilation of the HDL source code shown in **Figure 8A**.

**Figure 8C** is an example of the lower level of the RTL netlist which is derived from the lower level of the HDL source code shown in **Figure 8A**.

**Figure 8D** is an example of a technology netlist which is obtained from mapping of the RTL netlist which itself was derived from the HDL source code  
10 shown in **Figure 8A**. **Figure 8D** shows only a portion of the technology mapped netlist.

**Figure 9A** shows an example of a design of two integrated circuits before the replication of certain logic according to one aspect of the present invention.

**Figure 9B** illustrates the design of two integrated circuits after the replication  
15 according to one embodiment of one aspect of the present invention.

**Figure 9C** and **Figure 9D** illustrate another example of a replication of logic according to one embodiment of the present invention.

**Figures 9E** and **9F** show another example of a replication of logic between two floorplanned regions either on the same IC or two different ICs.

**Figure 10A** and **Figure 10B** illustrate an example of a method of splitting  
20 an RTL component according to one embodiment of one aspect of the present invention.

**Figure 10C** and **Figure 10D** illustrate another example of splitting an RTL component according to one embodiment of one aspect of the present invention.

**Figure 10E** shows an example of an embodiment of a method according to the present invention in which the splitting of an RTL component may be performed  
5 automatically.

**Figure 11A** and **Figure 11B** illustrate an example of a user interface for performing a floor planning operation.

**Figure 12** shows an example of an optimization process which may be performed as one aspect of an embodiment of the present invention.

10 **Figure 13** shows an example of machine readable media according to one embodiment of the present invention.

## DETAILED DESCRIPTION

Methods and apparatuses for designing an integrated circuit or a plurality of integrated circuits are described herein. In the following description, for purposes of explanation, numerous specific details are set forth in order to provide a thorough  
5 understanding of the present invention. It will be evident, however, to one skilled in the art that the present invention may be practiced without these specific details. In other instances, well-known structures, processes and devices are shown in block diagram form or are referred to in a summary manner in order to provide an explanation without undue detail.

10 Many of the methods of the present invention may be performed with a digital processing system, such as a conventional general purpose computer system. **Figure 2** illustrates a block diagram of a computer system that may be used to implement embodiments of the present invention. The computer system is used to perform logic synthesis of a design that is described in an HDL code. The computer system  
15 includes a processor 102 which is coupled through a bus 101 to a random access memory 104 and a read-only memory 106 and a mass storage device 107. Mass storage device 107 represents a persistent data storage device such as a floppy-disk drive, a fixed disk drive (e.g., magnetic drive, optical drive, or the like). Processor 102 may be embodied in a general purpose processor (such as the Intel Pentium®  
20 processors) a special purpose processor or a specially programmed logic device. Display 120 is coupled to the processor 102 through bus 101 and provides graphical output for the computer system. This graphical output is typically a graphical user interface which may be used to control the operation of the computer system.

Keyboard 121 and cursor control device 122 are coupled to bus 101 for communicating information and command selections to processor 102. The cursor control device 102 will typically be a mouse or other cursor control device which will be used to control a cursor displayed on the display device 120. Also coupled to  
5 processor 102 through bus 101 is an input/output interface 123 which can be used to control and transfer data to and from electrical devices such as printers and other computers which are coupled to the computer system 100.

It should be noted that the architecture of **Figure 2** is provided for purposes of illustration only and that a computer system or other digital processing system used  
10 in conjunction with the present invention is not limited to this specific architecture.

A general example of certain embodiments of the present invention will now be provided while referring to **Figure 3**. While most embodiments of the present invention are intended for use in HDL design synthesis software, the invention is not necessarily limited to such use. Although use of other languages in computer  
15 programs is possible, embodiments of the present invention will be described in the context of use in HDL synthesis systems, and particularly those designed for use with integrated circuits which have vendor specific technology/architectures.

As is well-known, the target architecture is typically determined by a supplier of programmable ICs. An example of a target architecture is the programmed look-up  
20 tables (LUTs) and associated logic of the Xilinx XC integrated circuits which is a field programmable gate array from Xilinx, Inc. of San Jose, California. Other examples of target architecture/technology include those well-known architectures in FPGAs and complex programmable logic devices from vendors such as Altera, Lucent

Technologies, Advanced Micro Devices, and Lattice Semiconductor. For certain embodiments, the present invention may also be employed with ASICs.

The method 201 of **Figure 3** begins in operation 203 in which a designer writes HDL source code which describes the desired circuitry/logic (e.g. a behavioral description of a desired circuit). This source code is then compiled by an HDL compiler in operation 205 to generate a technology independent RTL netlist. This netlist is independent of the particular vendor's technology which means that it is independent of the library of building blocks (e.g., look-up tables, multiplexers, AND/OR arrays, and so on) that is used by a particular target architecture. **Figure 8A** shows an example of HDL source code and **Figures 8B** and **8C** show an example to two levels of technology independent RTL netlists generated from the compilation of the source code shown in **Figure 8A**. In operation 207, a partition and/or a floor planning of the RTL netlist is performed. In the case of a partitioning, a portion of the RTL netlist is assigned to another integrated circuit. In the case of a floor planning, a portion of the RTL netlist is assigned to a particular region of an integrated circuit. In operation 209, the logic represented by the RTL netlist is optimized (e.g. the logic is optimized to improve timing parameters of the logic). This operation is optional and is used to remove duplicative interconnects and logic elements according to optimization routines. In operation 211, the RTL netlist (in either optimized or non-optimized form) is then mapped to the selected target architecture in order to generate the technology specific netlist. Then in operation 213, conventional place and route software tools are used to create a design of circuitry in the target architecture, such as a Xilinx or Altera FPGA.

The foregoing method shown in **Figure 3** will be understood to be one general example of a method of the present invention. As noted in operation 207, partitioning may be performed separately from floor planning in operation 207 such that only partitioning is performed in operation 207. Similarly, floor planning alone  
5 may be performed in operation 207. Alternatively, the combination of partitioning and floor planning may be performed in operation 207. Thus, operations relating to partitioning and/or floor planning are performed within an HDL synthesis process as opposed to before or after the synthesis is processed as in the prior art. This allows for, as described herein, improved interactivity in designing an integrated circuit and  
10 reduces the design time involved in designing an integrated circuit.

**Figures 4A and 4B** show specific examples of embodiments of the present invention for partitioning and floor planning respectively as separate aspects of the invention. However, as noted above, these two aspects may be used together in one embodiment of the present invention.

**Figure 4A** shows a method 301 in which partitioning of technology independent RTL netlists is performed between several integrated circuits. The method begins in operation 303 in which an HDL source code is prepared. Then in operation 305, the HDL source code is compiled to generate a technology independent RTL netlist. Optionally in operation 307, a hierarchical interconnect optimization is  
15 performed. In operation 309, an optional hierarchical resource estimation is performed. Operations 307 and 309 will be described further below. These operations improve the process of partitioning and thus are often desirable but not  
20 necessary.

The technology independent RTL netlist is typically a higher level behavioral representation of the design. This conserves abstracted information for use by processes before the final mapping step. This differs from traditional synthesis tools that fragment designs into fine, low level (gate) representations immediately after  
5 doing language compilation. By preserving a higher level behavioral representation, a synthesis tool can perform optimization, partitioning and floor planning at a much more global level and typically deliver better results. By operating on abstracted data, the synthesis tool can also operate more quickly and handle larger designs.

After the optional operations 307 and 309, operation 311 involves the  
10 performance of a partitioning of the technology independent RTL netlist between integrated circuits. This typically involves a selection by a user or by the system automatically of certain RTL modules which are assigned to different integrated circuits. This divides the design across several integrated circuits while remaining under the logic and input/output limitations of each integrated circuit. In one  
15 embodiment of the present invention which is described below, a user partitions the design by selecting blocks from the RTL block diagram presented on a display device and dragging them onto the target FPGA device. One embodiment of the invention provides immediate feedback on the display device on a percentage of input/output utilization and area utilization for each device. This gives the user considerable control  
20 over the assignment of RTL components to separate integrated circuits and gives quick feedback of the consequences of partitioning decisions. One embodiment of the invention provides an easy to use graphical user interface, which is described below, which allows dragging and dropping of RTL objects onto different integrated circuits.



This instructs the computer system to perform the partitioning operation 311 wherein the RTL netlist portions which were selected are partitioned among the integrated circuits as indicated. The hierarchical resource estimation operation 309 allows one embodiment of the invention to provide immediate feedback after this dragging and dropping operation. This feedback includes the percentage of input/output and area utilization for each integrated circuit and gives a user quick feedback of the consequences of a partitioning decision. Using the connectivity between the selected module which is dragged to a particular IC and all previously partitioned modules, one embodiment of the present invention computes the area and input/output impact of assigning the selected module to each of the FPGAs in a prototype system and feeds this information back to the user. The designer can then use the information to assign the selected module to the most appropriate FPGA while ensuring the resulting partition is within the requirements of the area of an integrated circuit and input/output utilization of a particular integrated circuit. When the partitioning operation is complete, this new arrangement of RTL netlists on different integrated circuits is then used to drive the final synthesis algorithms which typically map the RTL netlist to a particular target technology/architecture. It is noted that the choices of partitioning affect the circuit implementations so that the timing critical paths can be optimized while taking into account the delays associated with going on and off an integrated circuit and delays introduced by a printed circuit board. This is described further below.

Operation 311 may optionally include performing either a replication of logic from one integrated circuit to another integrated circuit or optionally splitting a unitary

RTL component into a first and second portion of the RTL component onto two integrated circuits. The replicating and splitting operations which are optional within operation 311 are described further below.

Following operation 311, an optional optimization of logic in the RTL netlist is performed in operation 313. This optimization may be a conventional optimization and typically includes the removal of duplicative logic components and interconnects. In operation 315, the RTL netlist is mapped to a particular target architecture in order to create a design of circuitry in the target architecture. In operation 317, conventional place and route software tools may then be used to implement circuitry in the target architecture.

**Figure 4B** shows a method 351 which includes a floor planning operation embedded within a synthesis according to an embodiment of the present invention. Operations 353, 355, 357, and 359 are similar to operations 303, 305, 307, and 309 of **Figure 4A**. In operation 361, floor planning is performed using the technology independent RTL netlist by allocating a portion of the RTL netlist to a portion of an integrated circuit. Further, as an optional operation, partitioning and/or replicating and/or splitting may also be performed. Operations relating to replicating and/or splitting are further described below. Floor planning at this stage of the synthesis process allows for manipulations based on individual registers, but floor planning is not overwhelmed by the gate level detail that follows technology mapping. Floor planning before technology mapping provides an opportunity to replicate logic and/or split logic for performance purposes. This capability can help correct routing problems, because global routing can be reduced by replicating objects that have a

small number of inputs and a large number of outputs. Performing floor planning at this stage of the synthesis process helps to reduce the number of iterations that are necessary to meet timing budgets. Another benefit will typically be the dramatic decrease in place and route processing times which are required in operation 367.

- 5 Furthermore, many FPGA vendors have mechanisms in their place and route software tools that allow floor planning information to be passed for netlist-level placement.

Following operation 361, the logic may optionally be optimized in operation 363. Then in operation 365, the RTL netlist is mapped to a particular target architecture to generate a technology specific netlist. Then the particular vendor's  
10 place and route software tool is used to create a design of circuitry in the target architecture.

Various aspects of the methods shown in **Figures 4A** and **4B** will now be described while referring **Figures 5A, 5B, 6, 7A, and 7B**. **Figure 5A** shows an example of a method for hierarchical interconnect optimization. The method 401  
15 includes operations 403 and 405. In operation 403, the interconnects between the RTL modules are examined. The RTL modules result from the HDL compilation process, such as operations 305 or 355 of **Figures 4A** and **4B** respectively. In operation 405, interconnect optimization is performed at the technology independent RTL netlist level. This typically will remove duplicative interconnects between the  
20 modules such as input/output connections between the modules. An example of this is shown in **Figure 5B**. Three HDL modules are shown in the assembly 411 of **Figure 5B**. HDL module 1 (at the RTL level) is shown as module 415. HDL module 2, shown as module 416 is coupled to module 1(module 415) by thirty

interconnect lines 424 and by the interconnect output 421. This output from module 1 originates from the AND gate 419 which includes two inputs, a constant value one, labeled as 418, and an output from the logic circuitry 420. Given that the constant value is one, the AND gate 419 will always pass the output value from logic circuitry 420 as the AND gate's output. Thus the output 421 will always be the same as the output 422. Thus these two outputs can be tied together to reduce one output from the RTL level module 415 shown in **Figure 5B**. Thus the operation 405 will identify these duplicative outputs or inputs to RTL level modules in the technology independent RTL netlist.

10           **Figure 6** shows a method for performing hierarchical resource estimation, such as operations 309 of **Figure 4A** and 359 of **Figure 4B**. The method 501 of **Figure 6** begins in operation 503 in which a user selects a target architecture. Then in operation 505, the system performs a mapping for the purpose of estimation of the RTL netlist for each module. This mapping is to the target architecture selected in  
15           operation 503. The system in operation 507 then specifies logic and input/output resource estimates based upon the selection of the target architecture and the mapping performed in operation 505. These logic and resource estimates are based upon a synthesis which is designed to estimate the logic requirements and input/output requirements of the particular design in order to implement the module in the target  
20           architecture. Further, in one embodiment, these estimates specify the timing estimates for each module after the mapping operation. In operation 509, the user may consider these resource and timing estimates which are displayed to a user in deciding how and

whether to partition and/or floor plan and/or replicate and/or split according to various aspects of the present invention.

**Figure 7A** shows a method 601 in which a user may perform the operation 311 in the case of partitioning. Further, the user may optionally perform the replicating and/or splitting operations or may perform a floor planning operation by assigning an RTL netlist module to the same or another integrated circuit. It will be appreciated that a partitioning and a floor planning may be combined in a series of operations in which a user selects one or more RTL netlist modules and indicates a placement on a different integrated circuit at a particular location or locations. After operation 603, operation 605 involves the replication of signals between the integrated circuits, such as input signals, and then in operation 607, the system creates a new RTL netlist for each integrated circuit.

**Figure 7B** shows an example of a user interface for performing a partitioning operation according to one aspect of the present invention. A display screen 605 is shown on a display device, such as a display device 120. A window 619 includes a representation of the available area of two integrated circuits. These areas 620 and 621 may be used to specify partitions or floor planning operations. A cursor 618 may be used by a user to perform a drag and drop operation from a representation of an netlist module, such as the RTL netlist modules 623 and 624 shown in the window 622. In a typical partitioning operation, the user uses the cursor 618 to select a particular netlist module which is then dragged to one or the other of the areas 620 or 621 to specify the particular integrated circuit which is intended to receive the design of circuitry created the selected netlist. Thus for example, the user may select the

netlist module 624 by positioning the cursor 618 over a region of the icon or image representing the netlist module 624 and by dragging the cursor after a selection of module 624 towards the area 620 representing integrated circuit one or the area 621 representing integrated circuit two within the window 619 thereby causing this netlist  
5 module to be partitioned to either integrated circuit.

Further aspects of the user interface as shown in **Figure 7B** will now be described by referring also to **Figures 8A** and **8B**. **Figure 8A** shows HDL source code at two levels, a top level 703 and a bottom level 705. The top level source code 703 specifies the RTL representation shown in **8B**, while the bottom level  
10 representation 705 specifies the RTL representation 713 shown in **Figure 8C**. The top level RTL representation shown **Figure 8B** includes two RTL modules 713 and 715 which in this case are identical and which are interconnected by interconnect 719. Input 711 feeds inputs to both RTL modules 713 and 715. The RTL module 715 provides an output 717. Thus, in one example, the RTL module 623 of **Figure 7B**  
15 represents the RTL module 713 of **Figure 8B** and the RTL module 624 of **Figure 7B** represents the RTL module 715 of **Figure 8B**. Thus, as shown in **Figure 7B**, the user may select either module and partition it to a second integrated circuit such as the representation of the integrated circuit 621 shown in **Figure 7B**. After the partitioning operation, a mapping operation is performed as described above resulting  
20 in a technology specific netlist, a portion of which is shown in **Figure 8D**.

**Figures 9A, 9B, 9C** and **9D** will now be referred to in describing one aspect of the present invention which involves replicating logic between several integrated circuits. This aspect may arise as a result of a partitioning as in the present

invention or without the partitioning of the present invention. **Figure 9A** includes two integrated circuits 803 and 805 which are interconnected by a 32 bit bus fed by outputs 819 as shown in **Figure 9A**. Integrated circuit 803 includes logic a and logic b labeled as logic 807 and logic 809. Further, a multiplexer 816 and a clocked register 817 are included within a logic block 815. Logic block 815 receives a 32 bit input and provides a 32 bit output to logic circuitry 809 and also to logic circuitry 811 and 813 in the integrated circuit 805 as shown in **Figure 9A**. An RTL representation of these two integrated circuits may display this level of detail to a user after compiling an HDL code to generate the RTL netlist before a mapping operation. The user may recognize that a replication of the logic block 815 is desirable because the input/output limitations of the integrated circuit 803 have been exceeded by the design shown in **Figure 9A**. This is in part due to the fact that a 32 bit input is required into the logic block 815 and a 32 bit output is required from the logic block 815 as shown in **Figure 9A**. The user may reduce the requirements for input/output pins on the integrated circuit 803 by replicating the logic block 815 onto the integrated circuit 805. The resulting replication is shown in **Figure 9B** in which the user has selected logic block 815 for replication causing the logic block 815 to be replicated onto the integrated circuit 805, now shown as the integrated circuit 805a with the replicated logic block 815a. Thus, the integrated circuit 803a has saved 32 outputs 819 as shown in **Figure 9B** while adding two outputs 819a. The integrated circuit 805a in this example still has 32 bits in this case coming from the input 819c and has also added two inputs 819b from the logic A 807 as shown in **Figure 9B**. Thus a minimal increase in input/output requirements for integrated circuit two as shown in **Figure 9B** results in a significant

decrease in input/output requirements for integrated circuit 803 as a result of the replication of the logic block 815.

Figures 9C and 9D show another example of a replication operation in which a four integrated circuit system 831 shown in Figure 9C is reduced to a three  
5 integrated circuit system shown in Figure 9D by replicating the logic 833 onto three different FPGA integrated circuits 835, 837 and 839. In this case, the integrated circuit 833 is a 4 to 16 decoder which receives 4 inputs and provides 16 outputs to the three different integrated circuits 835, 837 and 839. By replicating the logic in the decoder into the three different field programmable gate arrays 835, 837, and 839,  
10 three integrated circuits may result as shown in Figure 9D. In particular, three FPGAs 835a, 837a, and 839a may result by placing this decoder onto each of these three integrated circuits while increasing the inputs to each of these integrated circuits by four and the outputs from each of these integrated circuits by sixteen, assuming that each of the FPGAs can handle this increased input/output requirement. If the  
15 outputs from the decoders are not needed as an output from each IC, then there is a net decrease of 12 ( $16-4=12$ ) I/Os for each IC.

In another aspect of an embodiment of the present invention, a replication operation may be performed between floorplanned regions on the same IC or different ICs. Figures 9E and 9F show an example in which a register, originally designed  
20 for placement in one floorplanned region, is replicated to exist in two floorplanned regions. These two floorplanned regions may be on the same IC or different ICs. Figure 9E shows two floorplanned regions 852 and 854 which may be on the same IC 851. The floorplanned region 852 includes logic A (labeled as logic 854) and



register 856 which receives an input 857 and which provides an output 858 to logic A. The floorplanned region 853 includes logic B (labeled as logic 855) which receives an input through line 859 from output 858 of register 856 in floorplanned region 852. Due to timing problems (e.g. a delay in the line 859) it may be desirable to replicate  
5 the register 856 so that logic A and logic B receive their inputs at substantially the same time. The replication operation described herein may be performed between the two floorplanned regions, resulting in the system shown in **Figure 9F**. The register 856 has been replicated into the floorplanned region 853A which, as before, includes logic 855 and now also includes register 856A which is coupled to the same input 857  
10 as register 856.

**Figures 10A, 10B, 10C, 10D, and 10E** show another aspect of the present invention in which a single unitary RTL component is split into several RTL components among multiple (e.g. two or more) integrated circuits. A single unitary RTL component is a component in an RTL netlist. There are two kinds of such RTL  
15 components: (1) a simple RTL component which directly corresponds to an HDL language construct (AND; MULTIPLY; MUX, etc.) and (2) a higher level RTL component which is derived from a collection of simple RTL components and which is recognized by an HDL compiler as a RTL component (well known examples of such higher level RTL components include abstractions of components such as  
20 counters, ROMs, RAMs and State Machines, etc.). Either kind of such RTL component may be considered to be a single RTL component in an RTL netlist. In the examples shown in **Figures 10A and 10B**, a read only memory (ROM) is split into two RTL components for placement onto two different integrated circuits. The

initial design shown in **Figure 10A** includes a ROM specified by a single unitary RTL component 903. This ROM has a 12 bit input and a 100 bit data output. This ROM 903 is designed for placement on the integrated circuit 901 which also includes a logic circuitry 905 and logic circuitry 907 labeled as logic a and logic b respectively.

- 5 When the input /output requirements of all the circuitry shown in **Figure 10A** exceeds the input/output availability of a particular desired IC, then the designer may split the RTL component 903 into two components on two separate integrated circuits as shown in **Figure 10B**. This results in the integrated circuit 909 which includes the logic circuitry 905 and a portion of the ROM 903 shown as 903a in **Figure 10B**.
- 10 The integrated circuit 911 now includes the logic 907b which has been partitioned to the integrated circuit 911 according to an aspect of the present invention and the ROM 903 has been split onto the integrated circuit 911 to create the ROM portion 903b as shown in **Figure 10B**. In this way, the ROM 903 originally shown in **Figure 10A** now has been split between two integrated circuits.

- 15 **Figures 10C and 10D** show another example of a splitting operation in which an adder 925 is split into two adders between two integrated circuits 935 and 939. Prior to a splitting operation, the design is shown in **Figure 10C** on an integrated circuit 925. This integrated circuit includes the logic circuitry 923 and the adder 925. The adder 925 includes the input A which is a 32 bit input 927 and the
- 20 input B which is a 32 input 929. A carry-in input 931 is also received by the adder 925. A result output 933 is a 32 bit output. If these required inputs and outputs exceed the limitations of the integrated circuit 925, then the designer may split a single unitary RTL component, such as the adder 925 into two components on two separate

chips as shown in **Figure 10D**. In this case, two integrated circuits 935 and 939 are created. The integrated circuit 935 includes the logic 923 and also includes a portion of the adder 925a. The other portion of the adder 925b is now located on the integrated circuit 939. Effectively, half of the inputs to the adder are now allocated to  
5 the integrated circuit 935 and the other half of the inputs are allocated to the integrated circuit 939. Similarly, half of the outputs are obtained from the integrated circuit 935 and half of the outputs are obtained from the integrated circuit 939 for the adder.

**Figure 10E** shows an example of an automatic process for performing a splitting of a single RTL component. This method begins in step 951 in which a  
10 single large RTL component is selected from the technology independent RTL netlist. It will be appreciated that multiple such components may each be selected for a splitting operation. Then in operation 953, a large RTL component is split into several RTL components and then it is determined whether these components after the splitting can still remain on the same integrated circuit given the area limitations of the  
15 integrated circuit and given the input-output limitations of the integrated circuit. If after splitting the several integrated circuits can remain on the circuit then the decision is made to proceed to operation 957 in which the splitting is not allowed. If on the other hand the split components cannot remain on the same integrated circuit, the operation 955 is performed in which the large RTL component is split into several  
20 RTL components on different integrated circuits.

In another aspect of an embodiment of the present invention, a splitting operation may be performed between floorplanned regions on the same IC or different ICs. For example, if a ROM or adder is located in one of two floorplanned regions on

an IC, the ROM or adder may be split (as in the case of **Figures 10B or 10D** respectively) such that part of the ROM or adder may be located in one floorplanned region and another part may be located in another floorplanned region on the same IC. Splitting an RTL component between floorplan regions may result in a reduction of I/O (input/output) requirements. Furthermore, such splitting may be useful when a single RTL component cannot be fit into a floorplan region, which may occur in the case of certain FPGAs where regions, which may be allocated as a floorplanned region, are predetermined by the architecture of the FPGA. An example is the Altera Flex architecture which organizes available logic cells into rows.

**Figures 11A and 11B** will now be referred to while describing one embodiment of a floor planning method according to the present invention. This embodiment may use a graphical user interface within a window 1001 on a display device. This window 1001 includes a window 1003 which represents the available integrated circuit area and also includes areas which represent various RTL netlist modules, such as modules 1006, 1007, 1008, and 1009. A cursor 1005 which is movable under the control of the user is also shown on the display within the window 1001. The user may control the position and functionality of the cursor using a conventional cursor control device such as a mouse. A user may position the cursor 1005 over a particular module, and may select the module and drag and drop the module onto an available IC area. In this manner a user may perform a floor planning operation by specifying the placement at a particular location on one IC. When multiple ICs are available, multiple IC windows 1003 may also be displayed within the window 1001 for concurrent partitioning and/or floor planning operations.

**Figure 11B** shows an example of the result of a floor planning operation in which the user has allocated the available integrated circuit area within the window 1003 by performing certain operations or commands with a computer system. In one

and 1009 into the window 1003, and the system may automatically determine the available area required by each of the modules. In one embodiment of the invention, the resource estimation process in operations 309 and 359 may be used to determine the available IC area required for each of the modules. This allows the system to

5 automatically allocate a portion of the IC upon the user specifying that a particular module is to be placed at a certain position on the integrated circuit. As shown in **Figure 11B**, modules B and C have been combined into the region 1003c of the IC while module A has its own region 1003a, and module D has the area 1003b as shown in **Figure 11B**.

10 **Figure 12** shows one example in which a partitioned design may, after partitioning, utilize a known or estimated interchip or inter region delay 1025 in order to optimize the logic in the RTL netlist in order to meet system timing goals, such as in operation 209 shown in **Figure 3**. In particular, after a partitioning operation, two integrated circuits 1021 and 1023 may result from the partitioning. A known or

15 estimated interchip delay resulting from the delay of the interconnect on a printed circuit board, such as the interchip delay 1025 may be used in the optimization process, such as the process 209 or 313 described above. In particular, the interchip delay 1025 may be included in the timing constraints used in analyzing the timing from the input 1031 to the output 1032 between two integrated circuits 1021 and

20 1023. The input 1031 is first received by the clocked register 1029 which provides an input to the logic 1027 which outputs its signal through the interchip interconnect to the logic 1028 which then outputs its signal to the clocked register 1030 resulting in an output at the output 1032. Since the partitioning has occurred before the optimization

process (e.g. the partitioning of operation 311 occurs before the optimization process 313 as shown in **Figure 4A**) it is possible to include the timing delay resulting from the interchip delay 1025 in the timing constraints used during the optimization process.

5           One embodiment of the present invention may be a circuit design and synthesis computer aided design software that is implemented as a computer program which is stored in a machine readable media, such as a CD ROM or a magnetic hard disk or an optical disk or various other alternative storage devices. **Figure 13** shows an example of such media 1051 which includes in this case two partitions between what  
10   may be a volatile portion 1052 and a non volatile portion 1053. The volatile portion includes storage for source code and netlists which are compiled from the source code. In the example shown in **Figure 13**, a source code HDL file 1055 is stored in the memory 1052 and, after the compilation of the source code, a technology independent RTL netlist 1057 is stored in memory which is typically non volatile,  
15   although not necessarily non volatile in certain instances. In a typical implementation, the portion 1053 will be non volatile memory, such as a CD ROM or a magnetic hard disk which will retain the programming instructions necessary to perform the various processes of embodiments of the present invention, including the compilation of HDL source code, the technology mapping and optimization as well as the partitioning,  
20   floor planning, replication, and splitting operations as described above. Thus, for example, the storage media 1053 may include a HDL compiler routine specified in computer program instructions. This HDL compiler routine 1061 will operate upon an HDL source code, such as the HDL source code stored in memory as file 1055 in

order to produce the technology independent RTL list also stored in memory as netlist 1057. Optimization routines, such as optimization routines 1065 may also be stored on the machine readable media in order optimize the technology independent RTL netlist. Technology mapping routines 1063 are used to perform mapping operations  
5 from the technology independent netlist to a technology specific netlist as described above. Software routines which may perform various processes of the present invention are also stored in the machine readable media, including the partitioning routines 1067, the floor planning routines 1069, the replication routines 1071 and the splitting routines 1073.

10           The operations of the various methods of the present invention may be implemented by a processing unit in a digital processing system which executes sequences of computer program instructions which are stored in a memory which may be considered to be a machine readable storage media. The memory may be random access memory, read only memory, a persistent storage memory, such as mass  
15 storage device or any combination of these devices. Execution of the sequences of instruction causes the processing unit to perform operations according to the present invention. The instructions may be loaded into memory of the computer from a storage device or from one or more other digital processing systems (e.g. a server computer system) over a network connection. The instructions may be stored  
20 concurrently in several storage devices (e.g. DRAM and a hard disk, such as virtual memory). Consequently, the execution of these instructions may be performed directly by the processing unit. In other cases, the instructions may not be performed directly or they may not be directly executable by the processing unit. Under these

circumstances, the executions may be executed by causing the processor to execute an interpreter that interprets the instructions, or by causing the processor to execute instructions which convert the received instructions to instructions which can be directly executed by the processor. In other embodiments, hard wired circuitry may  
5 be used in place of or in combination with software instructions to implement the present invention. Thus, the present invention is not limited to any specific combination of hardware circuitry and software, nor to any particular source for the instructions executed by the computer or digital processing system.

Although the present invention has been described with reference to specific  
10 exemplary embodiments, it will be evident that various modifications and changes may be made to these embodiments without departing from the broader spirit and scope of the invention as set forth in the claims. Accordingly, the specification and drawings are to be regarded in an illustrative rather than a restrictive sense.